

## Лабораторная работа 5.

### Часть 1.

#### «Циклические конструкции. Итерационные алгоритмы. Цикл *while*»

**Цель работы:** изучить синтаксис циклической конструкции *while* языка *Python* для программирования итерационных алгоритмов, продемонстрировать возможности конструкции *while* на примере разработки интерактивных приложений.

#### Основное задание

Разработать интерактивную программу «*Try to Guess the Number*» («Попробуй угадать число»), которая эмулирует игру на отгадывание числа. Суть игры сводится к следующему: компьютер генерирует случайное число из диапазона, к примеру, от 1 до 100, а пользователь пытается отгадать число. При каждой попытке компьютер «подсказывает» игроку, как соотносится вариант игрока с загаданным компьютером числом: загаданное число больше или меньше указанного. Как только игрок отгадывает число, компьютер должен «поздравить» его с выводом на экран угаданного числа и количества затраченных игроком попыток. Далее компьютер может «предложить» повторно сыграть в игру или выйти.

Для универсальности можно добавить возможность выбора диапазона генерирования компьютером случайных чисел, а также задание ограничения на количество попыток. В случае, если игрок не укладывается в заданное количество попыток, программа должна выводить надпись «*Game Over*» из лабораторной работы 2.

#### Индивидуальное задание

В соответствии с заданием своего варианта выполнить задания из Приложения «В» «Циклические алгоритмы».

#### Дополнительное задание\*

1. Модернизировать программы, которые были разработаны в предыдущих лабораторных работах: «*ATM*» (банкомат) и «*Coin changer*» (разменный аппарат), «*Dice*» (игра в кости), симулятор пирожков с сюрпризом и «*Mood Sensor*» (датчик настроения)). Их нужно переписать таким образом, чтобы у пользователя была возможность повторного выполнения программы без выхода из неё.

2. Написать программу «*Heads or Tails?*» (Орёл или решка?), которая бы «подбрасывала» условно монету, к примеру, 1000 раз и сообщала, сколько раз выпал орёл, а сколько – решка.

3. Переработать программу из основного задания таким образом, чтобы пользователь загадывал число, а компьютер, используя оптимальный алгоритм, число отгадывал.

#### Требования к выполнению

1. Программы должны быть снабжены комментариями, в которых необходимо указать краткое предназначение программы, номер лабораторной работы и её название, версию программы, ФИО разработчика и дату разработки.

2. В отчете для каждой разработанной программы привести блок-схему либо пошаговое описание алгоритма.

3. В программах, где это необходимо, предусмотреть возможность её повторного выполнения, а также защиты от ввода некорректных данных (так называемую «защиту от дурака»).

4. Каждая программа должна быть снабжена дружелюбным и интуитивно понятным интерфейсом.

#### Контрольные вопросы

1. Для чего используются циклы? Что такое итерация?
2. Какие разновидности циклов существуют?
3. Описать *Python*-синтаксис цикла с предусловием *while*.
4. Чем является выражение после ключевого слова *while* – инициализацией, условием или обновлением?
5. Какова роль оператора *break* в теле цикла?

6. Какова роль оператора `continue` в теле цикла?
7. Какова роль оператора `pass` в теле цикла?
8. Может ли выражение после ключевого слова `while` содержать истинное значение или значения других типов данных?
9. Что такое бесконечный цикл? Когда он применяется? Привести пример кода организации диалога на тему завершения программы, либо повторного выполнения программы.
10. Если необходимо использовать вложенные циклы `while` для вывода элементов прямоугольной матрицы в виде строк и столбцов, какой из циклов будет печатать строки: внутренний или внешний?

## Цикл While

### ИНСТРУКЦИЯ WHILE

Одной из форм организации итераций в *Python* является инструкция `while`, общий синтаксис которой имеет вид:

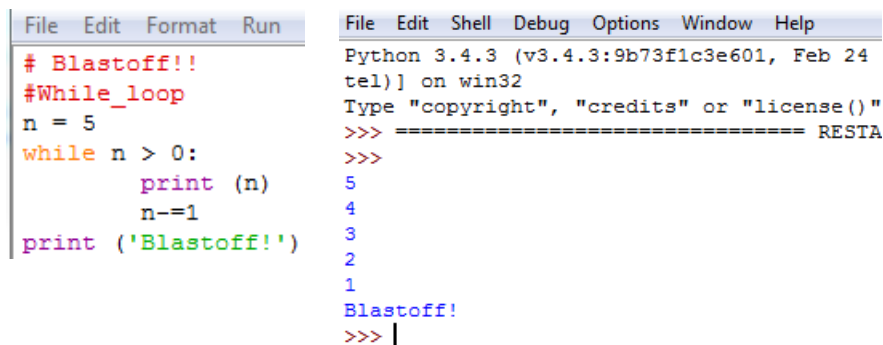
```
While condition:
    instruction 1
...
    instruction n
```

Поток выполнения для инструкции `while` имеет вид:

- 1) проверка условия (*condition*) (*True* или *False*);
- 2) если условие ложно, происходит выход из инструкции `while`, и выполнение продолжается со следующей инструкции;

3) если условие истинно, выполняется тело цикла (`instruction 1, ... , instruction n`) и происходит возврат к шагу 1. Такой поток называется циклом (*loop*), так как шаг 3 возвращается к началу алгоритма. Выполнение тела цикла называется *итерацией* (*iteration*). Итерации выполняются, пока условие истинно.

Код и результат программы, производящей обратный отсчет от 5 до 1 с выводом сообщения «*Blastoff!*», представлен на рисунке 5.1. Для цикла выполнено 5 итераций.



The screenshot shows a Python IDE with two panes. The left pane contains the following code:

```
# Blastoff!!
#While_loop
n = 5
while n > 0:
    print (n)
    n-=1
print ('Blastoff!')
```

The right pane shows the output of the program:

```
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24
tel)] on win32
Type "copyright", "credits" or "license()"
>>> ===== RESTART
>>>
5
4
3
2
1
Blastoff!
>>> |
```

Рис. 5.1. Код и результат программы, производящей обратный отсчет от 5 до 1 с выводом сообщения «*Blastoff!*»

Тело цикла должно изменять одну или более переменных, что в конечном итоге должно привести к ложности условия и завершению цикла. Переменные, которые изменяются каждый раз при выполнении цикла, и контролируют завершение цикла, называются *итерационными переменными* (*iteration variable*). Если итерационная переменная в цикле отсутствует, то такой цикл будет *бесконечным* (*infinite loop*).

**Замечание.** Обновление переменной путем прибавления к ней 1 называется *инкрементом* (*increment*), вычитание 1 называется *декрементом* (*decrement*).

### БЕСКОНЕЧНЫЕ ЦИКЛЫ И BREAK

Бесконечный цикл не содержит итерационную переменную, которая указывает на количество выполнений в цикле. Он используется в тех случаях, когда только при выполнении очередной итерации становится понятно, надо ли завершить цикл.

Инструкция `break` используется, когда нужно выйти из бесконечного цикла при наступлении заданного условия:

```
While True:
    instruction 1
...
    break
...
    instruction n
```

Если не добавить `break` в тело цикла, то произойдет «*зацикливание*» и цикл будет выполняться бесконечно.

Код и результат программы, которая бесконечно информирует пользователя, чему равна скорость света в вакууме, представлен на рисунках 5.2 – 5.4.

```
File Edit Format Run Options Window Help
while True:
    print ('Скорость света в вакууме равна 299792458 м/с!')
```

Рис. 5.2. Вывод бесконечного сообщения

```
File Edit Format Run Options Window Help
while True:
    x=float(input('Чему равна скорость света в вакууме (м/с)?\n'))
    if x!=299792458:
        print ('А вот и нет! Давайте попробуем еще раз!')
    else:
        print ('Отличные знания физики! (или умение искать в google:))')
        break
print('Поздравляем с правильным ответом')
```

Рисунок 5.3 – Вывод бесконечного сообщения с условными операторами

```
Чему равна скорость света в вакууме (м/с)?
56789777
А вот и нет! Давайте попробуем еще раз!
Чему равна скорость света в вакууме (м/с)?
59877755778
А вот и нет! Давайте попробуем еще раз!
Чему равна скорость света в вакууме (м/с)?
299792458
Отличные знания физики! (или умение искать в google:))
Поздравляем с правильным ответом
>>>
```

Рис. 5.4. Результат работы программы о скорости света в вакууме

## ЗАВЕРШЕНИЕ ИТЕРАЦИИ С ПОМОЩЬЮ CONTINUE

Инструкция `continue` пропускает текущую итерацию в цикле и переходит к следующей без завершения тела цикла. На рисунке 5.5 представлен цикл, который копирует данные, поступающие на вход до тех пор, пока не будет введено «*done*». Если строка начинается с символа «*#*», то она не выводится на печать.

```
while True:
    line = input('Введите строку\n')
    if line[0] == '#':
        continue
    if line == 'done':
        break
    print (line)
print ('Done!')
```

Рис. 5.5. Пример использования инструкции `continue`

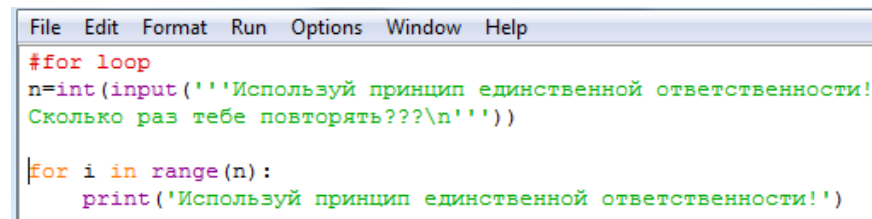
## Цикл *For*

Цикл `for` проходит через известное множество элементов столько раз, сколько элементов содержится во множестве. Пример кода и результата программы представлен на рисунке 5.6.

<pre>File Edit Format Run Options Window Help friends = ['Joseph', 'Glenn', 'Sally'] for friend in friends:     print ('Happy New Year:', friend) print ('Done!')</pre>	<pre>File Edit Shell Debug Opti Python 3.4.3 (v3.4.3:91 tel)] on win32 Type "copyright", "cre &gt;&gt;&gt; ===== &gt;&gt;&gt; Happy New Year: Joseph Happy New Year: Glenn Happy New Year: Sally Done! &gt;&gt;&gt;  </pre>
---	---

Рис. 5.6. Пример использования цикла `for`

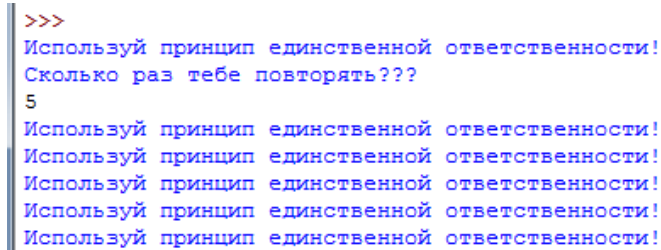
Конструкция `in range(n)` используется для цикла прохождения определенного числа шагов ( $n$ ) (рисунки 5.7, 5.8). Заметим, что итерационная переменная меняется от 0 до  $n - 1$ .



```
File Edit Format Run Options Window Help
#for loop
n=int(input('Используй принцип единственной ответственности!
Сколько раз тебе повторять???\n'))

for i in range(n):
    print('Используй принцип единственной ответственности!')
```

Рис. 5.7. Пример использования конструкции `in range(n)` в цикле `for`



```
>>>
Используй принцип единственной ответственности!
Сколько раз тебе повторять???
5
Используй принцип единственной ответственности!
Используй принцип единственной ответственности!
Используй принцип единственной ответственности!
Используй принцип единственной ответственности!
Используй принцип единственной ответственности!
```

Рис. 5.8. Результат выполнения конструкции `in range(n)` в цикле `for`

## ПРИЛОЖЕНИЕ В «Циклические алгоритмы»

**Задание 1.** Вычислить и вывести на экран в табличном виде значения функции  $F(x)$  в заданном диапазоне с некоторым шагом. Предусмотреть реагирование на случаи, когда параметры и (или) переменные принимают некорректные значения (например, значение аргумента не принадлежит области определения и др.)

$$F(x) = \begin{cases} -(kx^{k-1} + \sin(k+1)ab)^2 + (k+3)c, & x < 0, a \neq 0; \\ \frac{-a + kbx}{x+b} + 2\sqrt{c}, & x > 0, a = 0; \\ \frac{1}{kcb} + (k+1)^2 ab, & \text{в остальных случаях}; \end{cases}$$

Здесь  $x_1$  и  $x_2$  – значения аргумента, задающие диапазон изменения функции,  $dx$  – шаг изменения аргумента,  $F(x)$  – значение функции,  $a$ ,  $b$  и  $c$  – действительные числа, вводимые с клавиатуры,  $k$  – номер варианта. Результат получить в виде:

$x_1$	$x_2$	$dx$	$a$	$b$	$c$	$F(x)$

**Задание 2.** В соответствии с заданием своего варианта составить программу для вычисления значения функции с помощью разложения функции в степенной ряд. Задать точность вычислений  $eps$ . Предусмотреть максимальное количество итераций, равное 500. Вывести количество членов ряда, необходимых для достижения указанной точности вычислений. Результат получить в виде:

$x$	$n$	$F(x)$	$Math F(x)$	$eps$

Здесь  $x$  – значение аргумента,  $F(x)$  – значение функции,  $n$  – количество просуммированных членов ряда,  $Math F(x)$  – значение функции, вычисленное с помощью модуля *math*.

**Задание 3.** В соответствии с заданием своего варианта составить программу для нахождения суммы последовательности чисел.

Таблица В1. Индивидуальные задания.

Вариант	Задания	Условие
1	2	$\ln \frac{x+1}{x-1} = 2 \sum_{n=0}^{\infty} \frac{1}{(2n+1)x^{2n+1}} = 2\left(\frac{1}{x} + \frac{1}{3x^3} + \frac{1}{5x^5} + \dots\right),  x  > 1.$
	3	Организовать цикл, который принимает целые числа с клавиатуры и суммирует их квадраты. Окончание цикла – ввод числа 0.
2	2	$\arcsin x = \sum_{n=0}^{\infty} \frac{(2n)!}{4^n (n!)^2 (2n+1)} x^{2n+1} = x + \frac{x^3}{6} + \frac{3x^5}{40} + \dots,  x  < 1.$
	3	Организовать цикл, принимающий целые числа с клавиатуры и подсчитывающий количество отрицательных чисел. Окончание цикла – ввод числа, большего 100.
3	2	$\ln(1+x) = \sum_{n=0}^{\infty} (-1)^{n-1} \frac{x^n}{n} = x - \frac{x^2}{2} + \frac{x^3}{3} + \dots,  x  < 1.$
	3	Организовать цикл, принимающий целые числа и подсчитывающий количество положительных. Окончание – ввод 10.
4	2	$\ln(1-x) = \sum_{n=0}^{\infty} (-1) \frac{x^n}{n} = -x - \frac{x^2}{2} - \frac{x^3}{3} + \dots,  x  < 1.$
	3	Организовать цикл, который принимает целые числа с клавиатуры и подсчитывает количество четных чисел. Окончание цикла – ввод числа, большего 1000.

5	2	$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \dots$
	3	Организовать цикл, который принимает целые числа с клавиатуры и подсчитывает количество неотрицательных чисел. Окончание цикла – ввод числа, меньшего -100.
6	2	$\sin x = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!} = x - \frac{x^3}{3!} + \frac{x^5}{5!} + \dots$
	3	Организовать цикл, который принимает целые числа и вычисляет их среднее арифметическое. Окончание – ввод 0.
7	2	$\cos x = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n}}{(2n)!} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} + \dots$
	3	Организовать цикл, который принимает целые числа с клавиатуры и подсчитывает количество чисел, меньших числа 10. Окончание цикла – ввод числа 100.
8	2	$\arcsin x = \sum_{n=0}^{\infty} \frac{(2n)!}{4^n (n!)^2 (2n+1)} x^{2n+1} = x + \frac{x^3}{6} + \frac{3x^5}{40} + \dots,  x  \leq 1.$
	3	Организовать цикл, который принимает целые числа и суммирует каждое второе из них. Окончание цикла – ввод числа 0.
9	2	$\arccos x = \frac{\pi}{2} - \arcsin x = \frac{\pi}{2} - \sum_{n=0}^{\infty} \frac{(2n)!}{4^n (n!)^2 (2n+1)} x^{2n+1} = \frac{\pi}{2} - x - \frac{x^3}{6} - \dots,  x  \leq 1.$
	3	Организовать цикл, который принимает целые числа и вычисляет среднее арифметическое четных чисел. Окончание – ввод 0.
10	2	$\frac{1}{1-x} = \sum_{n=0}^{\infty} x^n = 1 + x + x^2 + \dots,  x  < 1.$
	3	Организовать цикл, который принимает целые числа с и вычисляет наибольшее из них. Окончание цикла – ввод числа 0.
11	2	$\ln \frac{x+1}{x-1} = 2 \sum_{n=0}^{\infty} \frac{1}{(2n+1)x^{2n+1}} = 2\left(\frac{1}{x} + \frac{1}{3x^3} + \frac{1}{5x^5} + \dots\right),  x  > 1.$
	3	Организовать цикл, который принимает целые числа и вычисляет наименьшее из них. Окончание цикла – ввод числа 0.
12	2	$\arcsin x = \sum_{n=0}^{\infty} \frac{(2n)!}{4^n (n!)^2 (2n+1)} x^{2n+1} = x + \frac{x^3}{6} + \frac{3x^5}{40} + \dots,  x  < 1.$
	3	Организовать цикл, который принимает целые числа и вычисляет количество чисел, больше 12. Окончание цикла – ввод 0.
13	2	$\ln(1+x) = \sum_{n=0}^{\infty} (-1)^{n-1} \frac{x^n}{n} = x - \frac{x^2}{2} + \frac{x^3}{3} + \dots,  x  < 1.$
	3	Организовать цикл, который принимает целые числа и вычисляет количество натуральных чисел. Окончание цикла – ввод 0.
14	2	$\ln(1-x) = \sum_{n=0}^{\infty} (-1) \frac{x^n}{n} = -x - \frac{x^2}{2} - \frac{x^3}{3} + \dots,  x  < 1.$
	3	Организовать цикл, который принимает целые числа и вычитает их из 10000. Окончание – получение отрицательного итога.
15	2	$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \dots$
	3	Организовать цикл, который принимает целые числа и суммирует их. Окончание цикла – получение числа, большего 100.
16	2	$\sin x = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!} = x - \frac{x^3}{3!} + \frac{x^5}{5!} + \dots$
	3	Организовать цикл, который принимает целые числа и вычисляет количество четных натуральных чисел. Окончание – ввод 0.
17	2	$\cos x = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n}}{(2n)!} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} + \dots$
	3	Организовать цикл, который принимает целые числа и вычисляет количество нечетных натуральных чисел. Окончание – ввод 0.

18	2	$\arcsin x = \sum_{n=0}^{\infty} \frac{(2n)!}{4^n (n!)^2 (2n+1)} x^{2n+1} = x + \frac{x^3}{6} + \frac{3x^5}{40} + \dots,  x  \leq 1.$
	3	Организовать цикл, который принимает целые числа с клавиатуры и суммирует их последние цифры. Окончание – ввод 18.
19	2	$\arccos x = \frac{\pi}{2} - \arcsin x = \frac{\pi}{2} - \sum_{n=0}^{\infty} \frac{(2n)!}{4^n (n!)^2 (2n+1)} x^{2n+1} = \frac{\pi}{2} - x - \frac{x^3}{6} - \frac{3x^5}{40} + \dots,  x  \leq 1.$
	3	Организовать цикл, который принимает целые числа и умножает их последние цифры. Окончание цикла – ввод числа 0.
20	2	$\frac{1}{1-x} = \sum_{n=0}^{\infty} x^n = 1 + x + x^2 + \dots,  x  < 1.$
	3	Организовать цикл, который принимает целые числа и суммирует их. Окончание цикла – ввод отрицательного числа.
21	2	$\ln \frac{x+1}{x-1} = 2 \sum_{n=0}^{\infty} \frac{1}{(2n+1)x^{2n+1}} = 2\left(\frac{1}{x} + \frac{1}{3x^3} + \frac{1}{5x^5} + \dots\right),  x  > 1.$
	3	Организовать цикл, который принимает целые числа и умножает их. Окончание цикла – ввод положительного числа.
22	2	$\arcsin x = \sum_{n=0}^{\infty} \frac{(2n)!}{4^n (n!)^2 (2n+1)} x^{2n+1} = x + \frac{x^3}{6} + \frac{3x^5}{40} + \dots,  x  < 1.$
	3	Организовать цикл, принимающий числа и суммирующий их кубы. Окончание цикла – ввод числа 12.
23	2	$\ln(1+x) = \sum_{n=0}^{\infty} (-1)^{n-1} \frac{x^n}{n} = x - \frac{x^2}{2} + \frac{x^3}{3} + \dots,  x  < 1.$
	3	Организовать цикл, который принимает целые числа и находит количество чисел, больших числа 23. Окончание – ввод 15.
24	2	$\ln(1-x) = \sum_{n=0}^{\infty} (-1) \frac{x^n}{n} = -x - \frac{x^2}{2} - \frac{x^3}{3} + \dots,  x  < 1.$
	3	Организовать цикл, принимающий целые числа и находящий количество попавших от 5 до 25. Окончание – ввод числа 0.
25	2	$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \dots$
	3	Организовать цикл, который принимает целые числа и находит количество попавших от 25 до 625. Окончание – ввод числа 0.



## Лабораторная работа 5.

### Часть 2.

#### «Циклические конструкции. Итерационные алгоритмы. Цикл *for*»

**Цель работы:** изучить синтаксис циклической конструкции *for* языка *Python* для программирования итерационных алгоритмов, продемонстрировать возможности циклических конструкций *while* и *for* на примере разработке интерактивных приложений.

#### Основное задание

Разработать программы для решения следующих задач, которые должен уметь реализовать каждый программист (*it's easy...*):

1. Найти сумму цифр и количество цифр заданного натурального числа.
2. Возвести число в натуральную степень  $n$ .
3. Найти количество различных цифр у заданного натурального числа
4. Найти наибольшую цифру натурального числа.
5. Задано натуральное число. Проверить, является ли заданное натуральное число палиндромом.
6. Определить является ли заданное натуральное число простым.
7. Найти все простые делители заданного натурального числа.
8. Найти НОД и НОК двух натуральных чисел.
9. Заданы три целых числа, которые задают некоторую дату. Определить дату следующего дня.
10. Запрограммировать последовательность чисел Фибоначчи (пользователь вводит порядковый номер элемента последовательности Фибоначчи, а программа выводит на экран значение).

#### Дополнительное задание\*

1. Совершенным называется число, равное сумме всех своих делителей, не равных самому числу, учитывая и 1. Проверить является ли заданное число совершенным.
2. Задано число, содержащее от двух и более цифр. Между каждой парой соседних цифр, вставить заданное число. Например, число 7: 243  $\rightarrow$  27473.
3. Задано натуральное число  $N$ . Каждое вхождение наибольшей цифры, использованной в записи числа  $N$ , продублировать. Например, 349291  $\rightarrow$  34992991.

#### Требования к выполнению

1. Программа должна обязательно быть снабжена комментариями на английском языке, в которых необходимо указать краткое предназначение программы, номер лабораторной работы и её название, версию программы, ФИО разработчика, номер группы и дату разработки.
2. В отчете для каждой разработанной программы привести блок-схему либо пошаговое описание алгоритма.
3. В программах, где это необходимо, предусмотреть возможность её повторного выполнения и защиту от ввода некорректных данных (предусмотреть так называемую «защиту от дурака»).
4. Для реализации всех программ использовать только числовые типы данных *Python* (*int*, *long*, *float*, *bool*) и строки (*str*).
5. Каждая программа должна быть снабжена дружелюбным и интуитивно понятным интерфейсом.

#### Контрольные вопросы

1. Описать *Python*-синтаксис и принцип работы цикла *for*.
2. В чём особенность использования циклической конструкции *for* в языке *Python* по сравнению с использованием в других языках программирования, к примеру, в *Pascal* или *C/C++*?
3. Для чего используется стандартная функция `range(...)`?
4. Описать наиболее распространённые случаи использования функции `range(...)`.
5. Описать сочетание функции `range(...)` и цикла *for*.
6. Обязательно ли использовать переменную цикла *for* внутри самого цикла?

7. Какую дополнительную конструкцию (ветку) могут иметь циклы в языке *Python*? В каких случаях тело данной конструкции будет выполняться, а в каких – нет?

